Constructive Meta-Level Feature Selection Method based on Method Repositories

Hidenao Abe
Shimane University, JAPAN
abe@med.shimane-u.ac.jp

Takahira Yamaguchi
Keio University, JAPAN
yamaguti@ae.keio.ac.jp

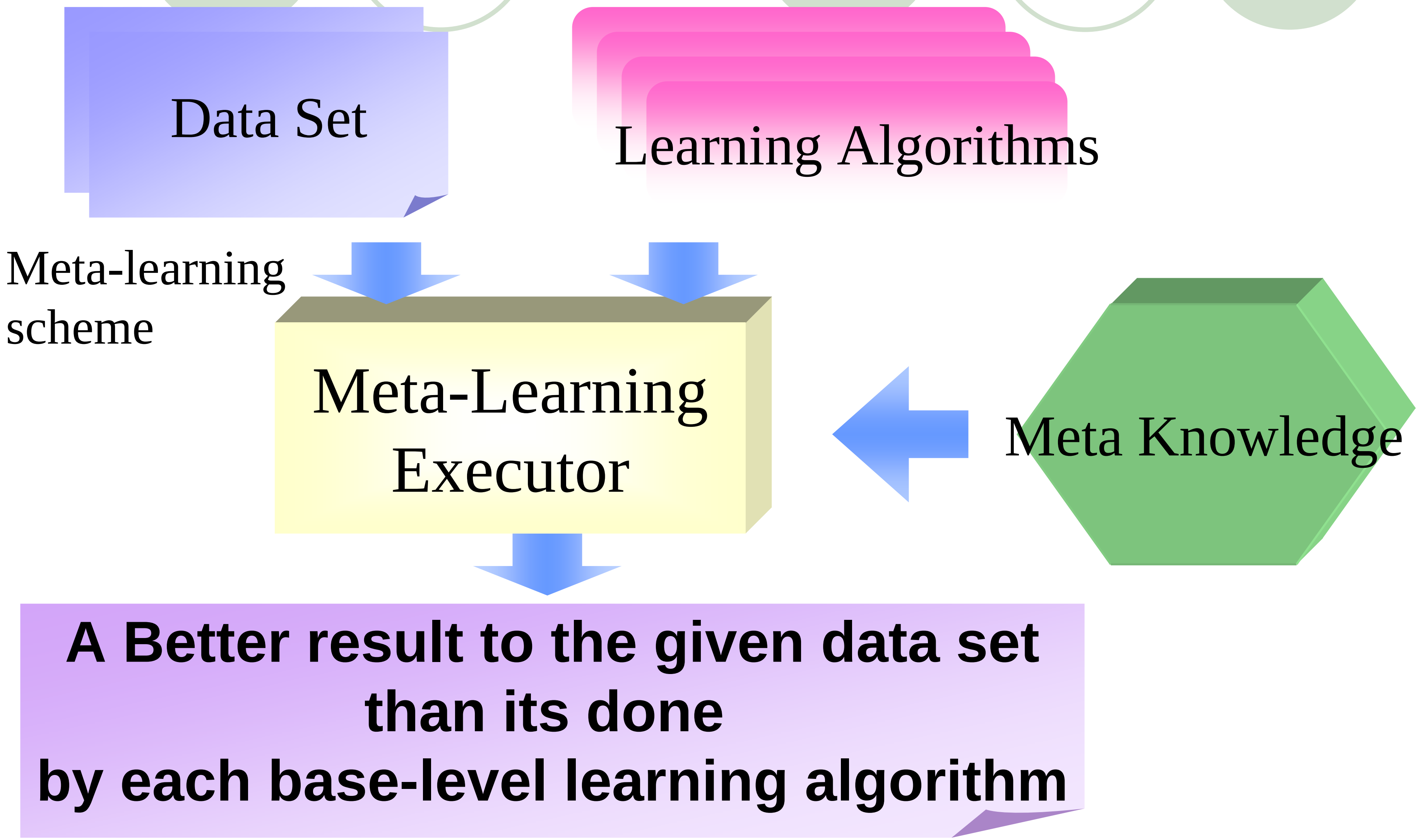
Contents

- Background
- Constructive meta-learning based on method repositories
- Experiment with common data sets
- Conclusion

Feature Selection Algorithms

- Filter Approach
  - Fast execution with low performance
- Wrapper Approach
  - Slow execution with high performance
  - Kind of search problem
    - However, to determine starting subset is not considered as a component of these algorithms
- Problem
  - How to choose the proper feature selection algorithm (FSA) to a given dataset, according to a user requirement

Overview of meta-learning scheme



Selective meta-learning scheme and our motivation

- Integrating base-level classifiers, which are learned with different training data sets generating by
  - “Bootstrap Sampling” (bagging)
  - weighting ill-classified instances (boosting)
- Integrating base-level classifiers, which are learned from different learning algorithms
  - simple voting (voting)
  - constructing meta-level classifier with a meta-level training data set (stacking, cascading)

They don’t work well, when no base-level algorithm works well to the given data set !!

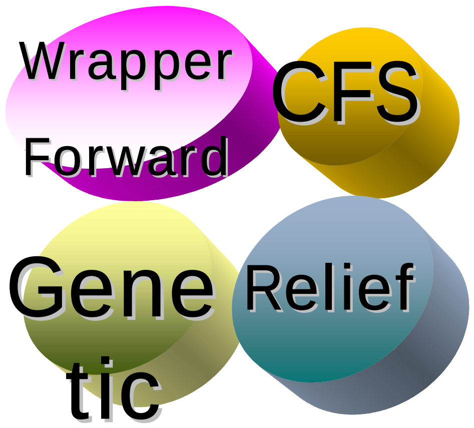
-> It is time to de-compose base-level algorithms and re-construct a proper algorithm to the given data set.

Contents

- Background
- Constructive meta-learning based on method repositories
  - Basic idea of our constructive meta-level feature selection
  - An implementation of the constructive meta-level feature selection
- Experiment with common data sets
- Conclusion

Basic Idea of our Constructive Meta-Level Feature Selection

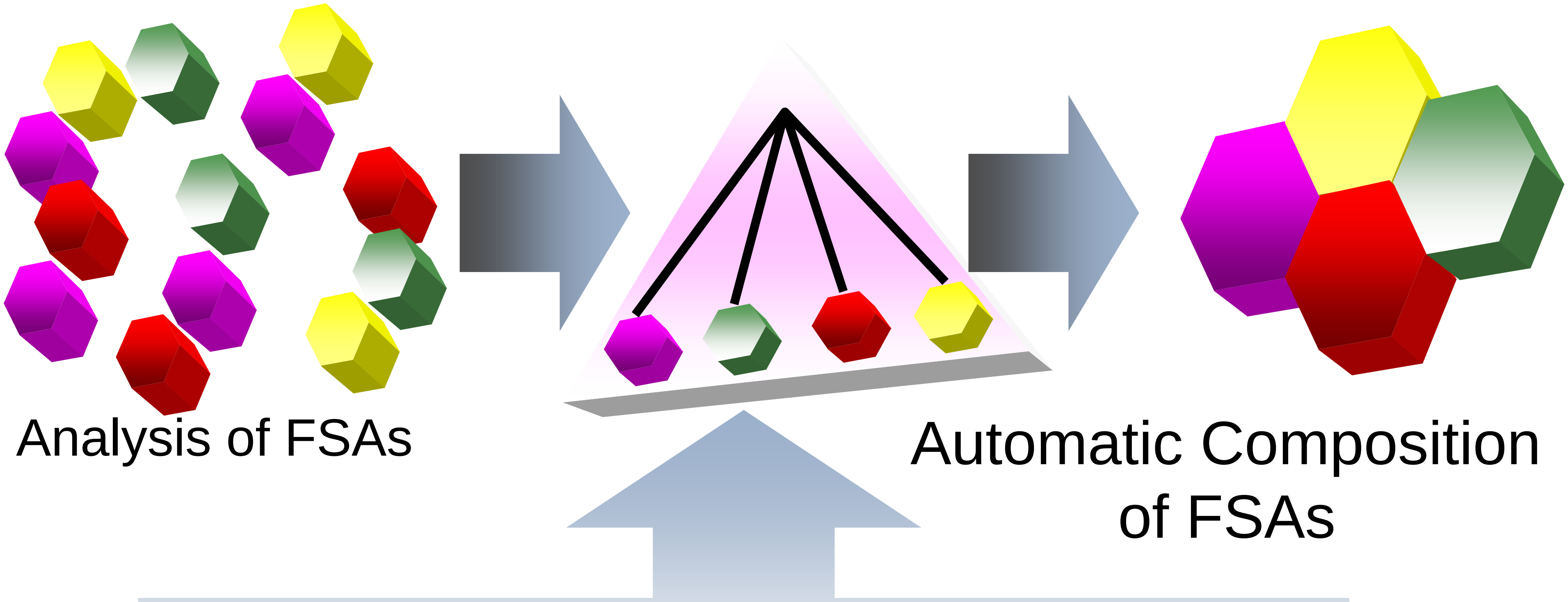
De-composition& Organization + Search & Composition



Analysis of FSAs

Basic Idea of our Constructive Meta-Level Feature Selection

De-composition& Organization + Search & Composition



Organizing feature selection methods, treated objects and control structures

Issues to implement meta-level feature selection method

- **<u>How to de-compose FSAs into methods (FSMs)</u>**
  - We de-composed FSAs in Weka Attribute Selection package in to four generic methods, according to their nature
- **<u>How to restrict combinations between methods to re-construct FSAs</u>**
  - We have described restrictions on input, output, reference, pre-method and post-method for each method. Then they have been organized as method hierarchy and data type hierarchy.
- **<u>How to re-costruct a proper FSAs to given dataset</u>**
  - We have developed a system to search for a proper FSA to a given dataset with the method repository

Contents

- Background
- Constructive meta-level feature selection based on method repositories
  - Basic idea of our constructive meta-level feature selection
  - An implementation of the constructive meta-level feature selection
- Experiment with common data sets
- Conclusion

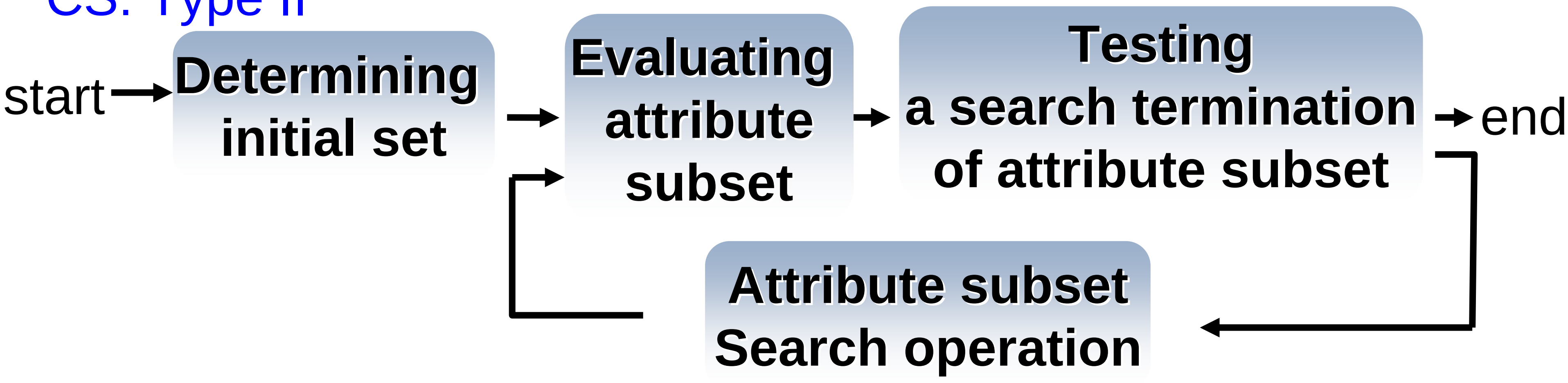
Analysis of FSAs

- Analyzing FSAs implemented in Weka
- Identified the four generic methods based on ‘search problem’
  - Determining initial set
  - Evaluating attribute subset
  - Testing a search termination of attribute subset
  - Attribute subset search operation
- Described restrictions of connections between two of the generic methods

Identifying FSAs Control Structures

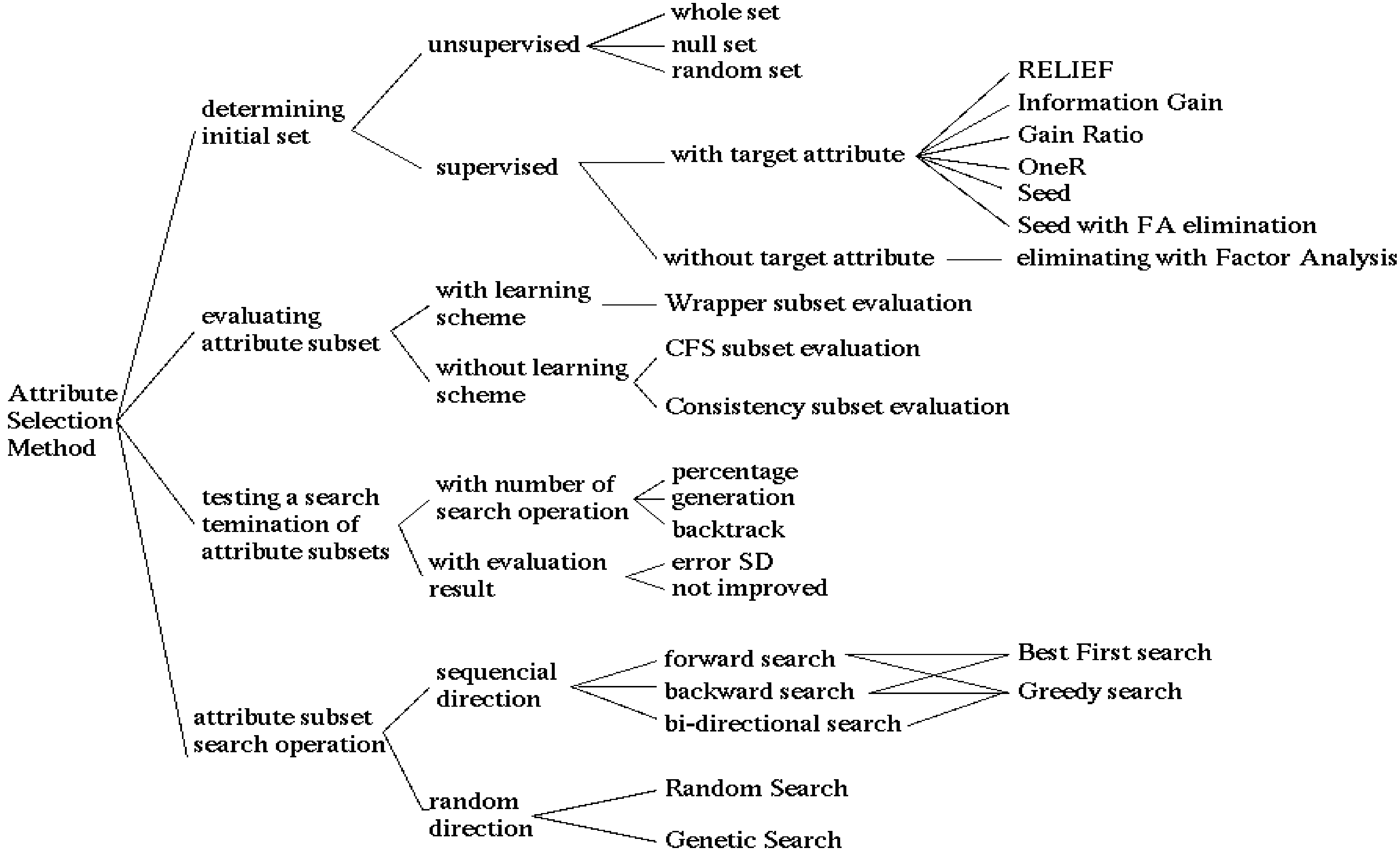
CS: Type I

CS: Type II



Type I: filter approach algorithms
Type II: wrapper and hybrid algorithms

Feature Selection Method Repository



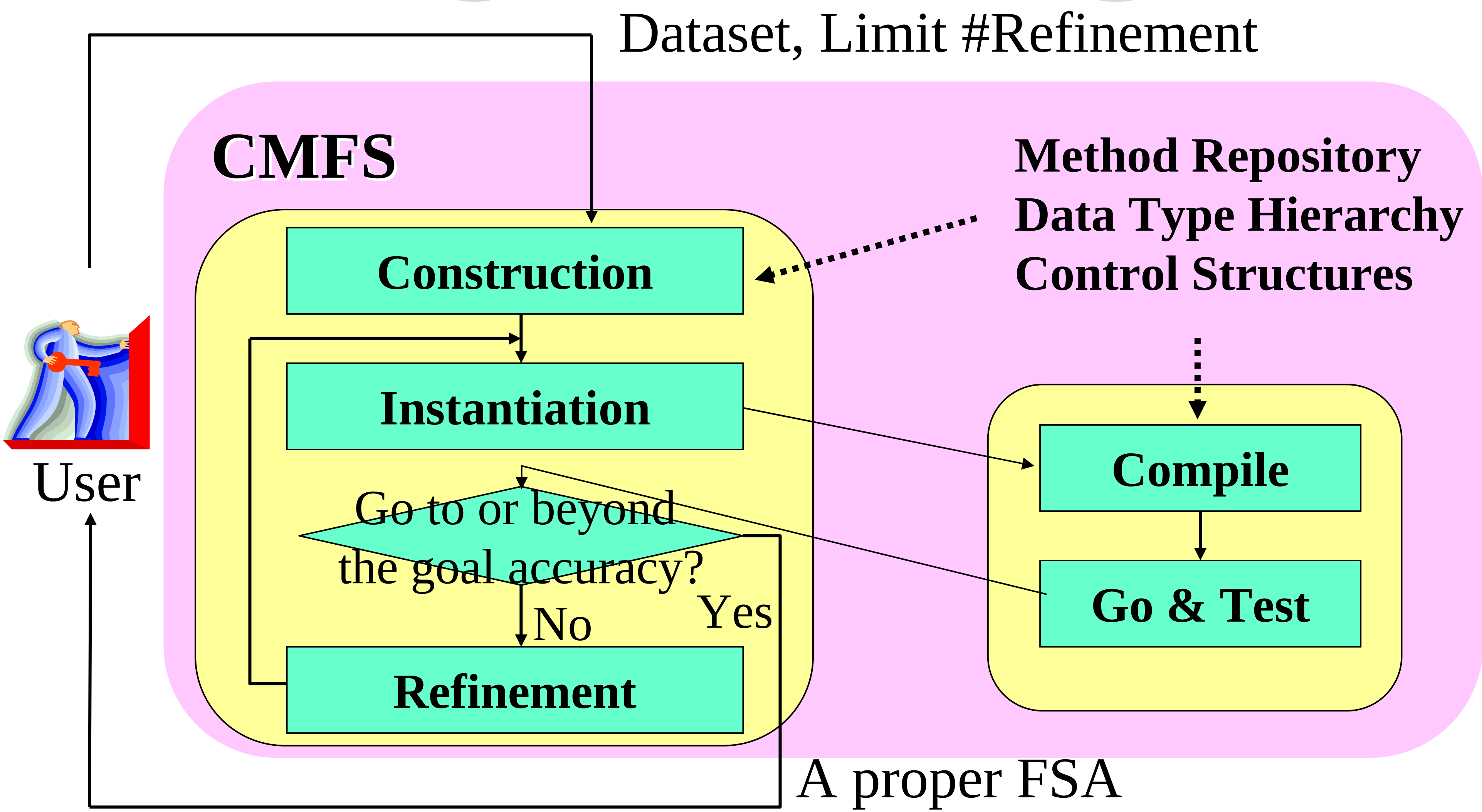
Data Type Hierarchy

Organization of input/output/reference data types for feature selection methods

- Objects
  - dataset
    - training dataset
    - validation dataset
    - test dataset
  - attribute-set
    - Whole (given) attribute set
    - Attribute subset
    - Null attribute set
  - attribute
    - Nominal attribute
    - Numerical attribute

System overview of CMFS:

a Constructive Meta-level Feature Selection tool



Contents

- Introduction
- Constructive meta-level feature selection based on method repositories
- Experiments: Accuracy comparison using UCI common data sets
- Conclusion

Experiment with UCI Common Datasets

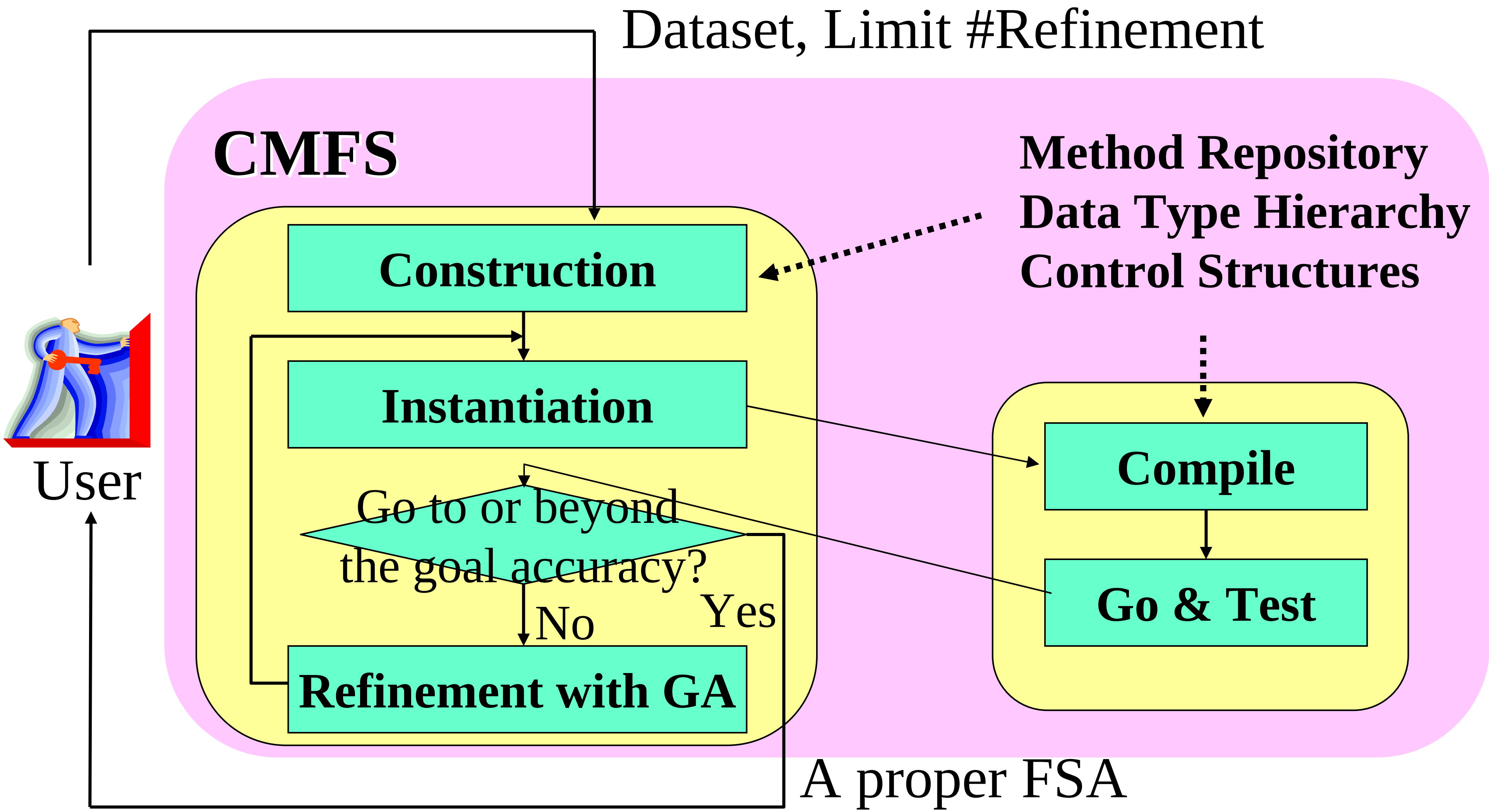
- Input: 32 UCI common datasets
- Comparison:
  - No feature selection
  - Seed initial subset determination + Forward selection
  - Genetic Search [Vafaie 92]
  - FSA constructed by CMFS
- Process:
  1. Select attribute subset with each FSA on each whole training dataset
  2. Carry out 10-fold CV with the datasets which have each attribute subset
  3. Compare averaged predictive accuracies among the FSAs

CMFS setting

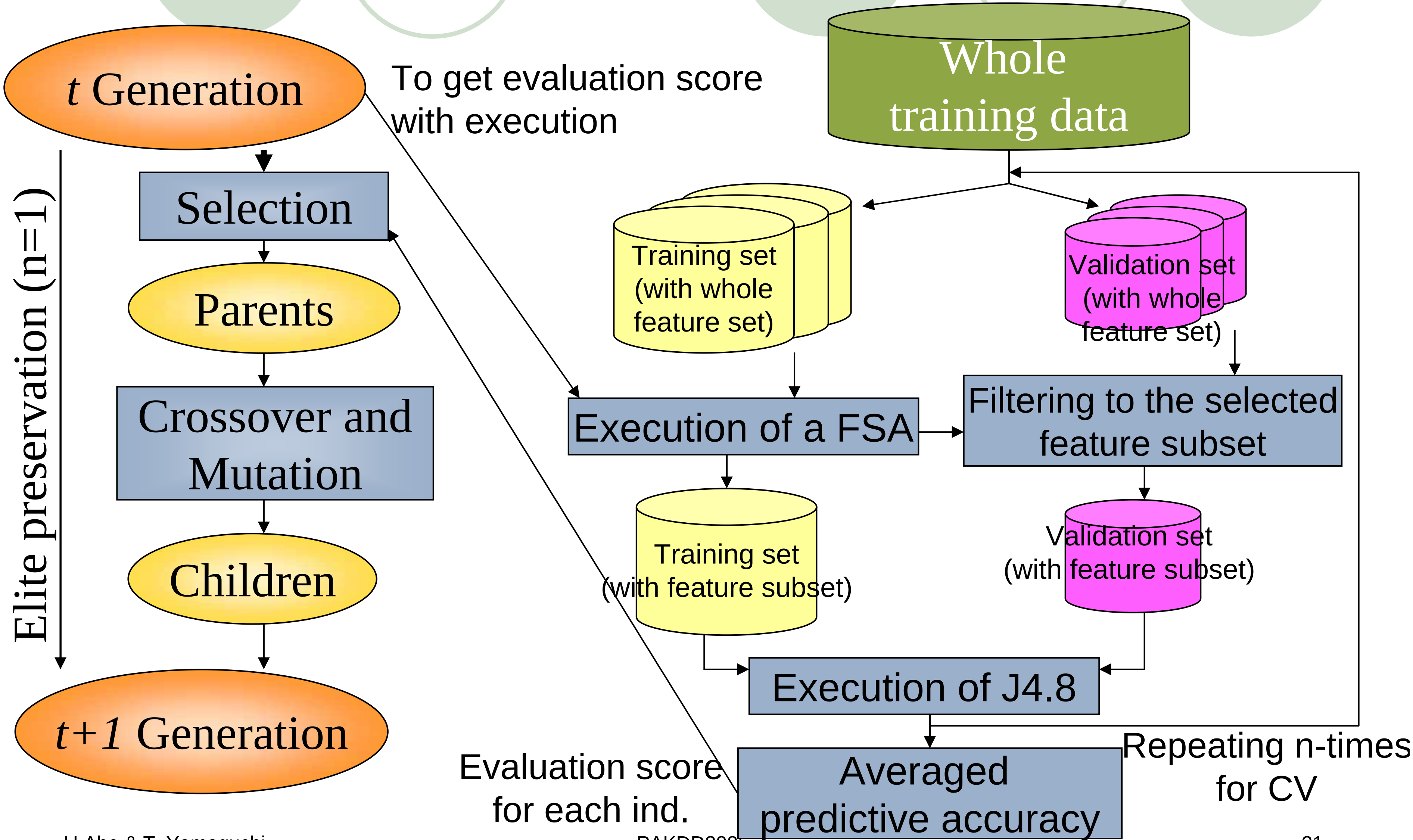
- CMFS has output just one specification of the composed FSA to each data set.
  - CMFS has searched 292 FSAs for the best one, executing up to one hundred FSAs.

- Search method in 'Refinement' is based on GA
  - each generation has 10 individuals
  - evaluating each individuals with alternative predictive accuracy
  - roulette selection with elite preservation (parents size = 6)
  - crossover on randomized single point
  - mutation at least one child (mutation probability=0.02)

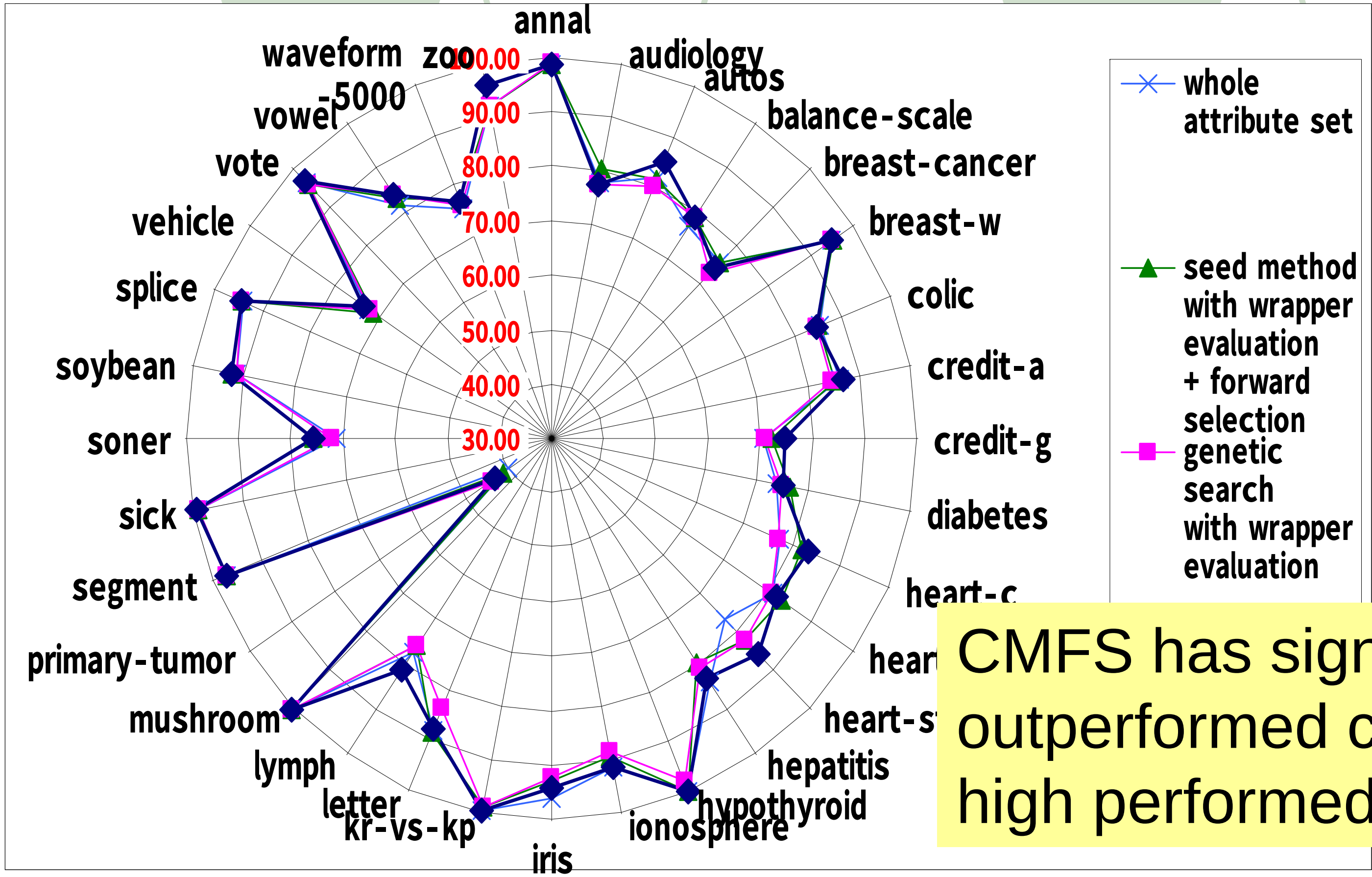
System overview of CMFS:
a Constructive Meta-level Feature Selection tool



Choosing a best FSA with GA refinement



Evaluation on 10CV accuracies of CMFS and three other feature subsets



Accuracy(%)= (#correctly classified/ #total test instances) *100

CMFS has significantly (p<0.05) outperformed compared to other high performed FSAs.

	Whole Att. set	Seed method	Genetic	CMFS
Average	84.41	84.86	84.08	85.76
#Best Acc.	7	9	5	17

New FSA, combining the FSMs
for heart-statlog

```
Input: Whole feature set  F, training data set Tr
Output: Feature subset for the training data set  Fsub
Prameters: number of backtracks=5

begin:
  Feature set f;
  f = determining_initial_set_with_FA+Seed(F);
  int i=0;
  double[] evaluations;
  while(1){
    evaluations[] = feature_subset_evaluation_with_CFS(f);
    (f,i) = backward_elimination(evaluations,f);
    if(number_of_backtracks(i,5)==true){ break; }
  }
  return f;
end:
```

Note: This FSA is automatically constructed from the FSM repository with CMFS.

Contents

- Introduction
- Constructive meta-level feature selection based on method repositories
- Case Study with common data sets
- Conclusion

Conclusion & Future Work

- CMFS has been implemented as a tool for “Constructive Meta-Level Feature Selection” scheme based on method repositories.
- FSAs constructed by CMFS have outperformed significantly, comparing with two high-performance FSAs.
- CMFS can construct proper FSAs to almost given datasets automatically.

- Feature work
  - Extending FSM repository
  - Combining constructive meta-learning scheme to construct a proper ‘mining application’ for a given dataset

Acknowledgement

- This research is supported by the Grant-in-Aid for Young Scientists(B), 17700152, by Ministry of Education, Science and Culture for Japan